PROIECTAREA ALGORITMILOR

Lect. univ. dr. Adrian Runceanu

Curs 6

Elemente de teoria grafurilor

Conţinutul cursului

6.1. Definiţii

6.2. Memorarea(reprezentarea) grafurilor

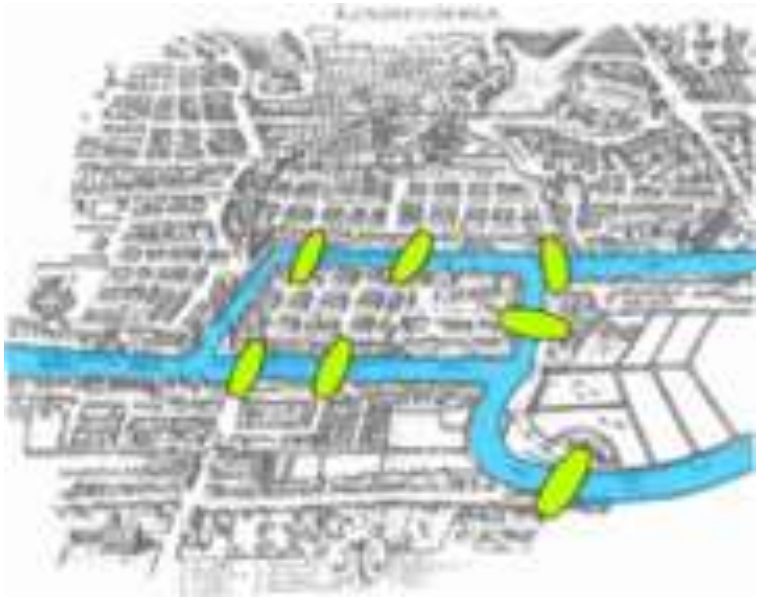
6.3. Parcurgerea grafurilor

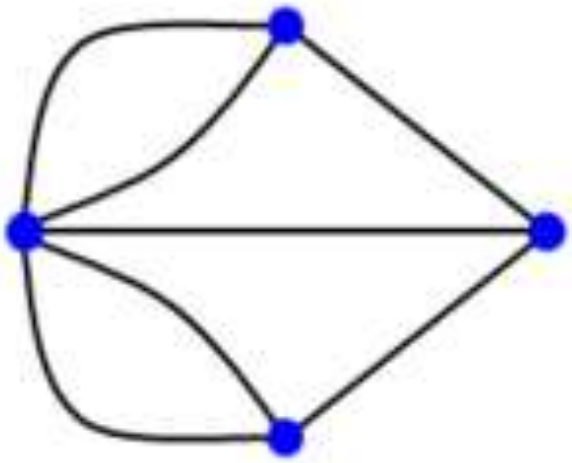
Scurte consideraţii istorice

- Originile teoriei grafurilor se găsesc în rezolvarea unor probleme de jocuri şi amuzamente matematice, care au atras atenţia unor matematicieni de seamă, cum ar fi:
  - ***Euler***
  - ***Hamilton***
  - ***Cazlyley***
  - ***Sylvester***
  - ***Birkoff***
- Data naşterii teoriei grafurilor este considerată a fi anul **1736,** când matematicianul ***Leonhard Euler*** a publicat un articol în care a clarificat ***problema celor şapte poduri*** şi a prezentat o metodă pentru rezolvarea altor probleme de acelaşi tip.

Scurte consideraţii istorice

- În oraşul Königsberg (Kalingrad) existau peste râul Pregel şapte poduri.
- Problema celor şapte poduri era:





- *Se poate face o plimbare peste toate cele şapte poduri, trecând o singură dată peste fiecare pod?*

6.1. Definiţii

- În scopul descrierii unor activităţi din cadrul unui proces de producţie sau a relaţiilor existente între elementele unei structuri organizatorice se pot folosi **imagini grafice** gen **diagrame**, **schiţe**, **grafice**, etc.

- O reprezentare dintre cele mai utilizate este cea prin **grafuri**.

- Acestea sunt utilizate în special pentru *vizualizarea sistemelor şi situaţiilor complexe*.

6.1. Definiţii

În general, vom reprezenta:

- componentele acestora prin **puncte** în plan

- relaţiile (legăturile, dependenţele, influenţele, etc) dintre componente prin **arce de curbă** cu extremităţile în punctele corespunzătoare.

6.1. Definiţii

Între două puncte pot exista:

- *unul sau mai multe segmente* (în funcţie de câte relaţii dintre acestea, care ne interesează, există)

- iar segmentelor:
  - li se pot asocia sau nu orientări (după cum se influenţează cele două componente între ele),
  - numere care să exprime intensitatea relaţiilor dintre componente etc.

6.1. Definiţii

Definiţie

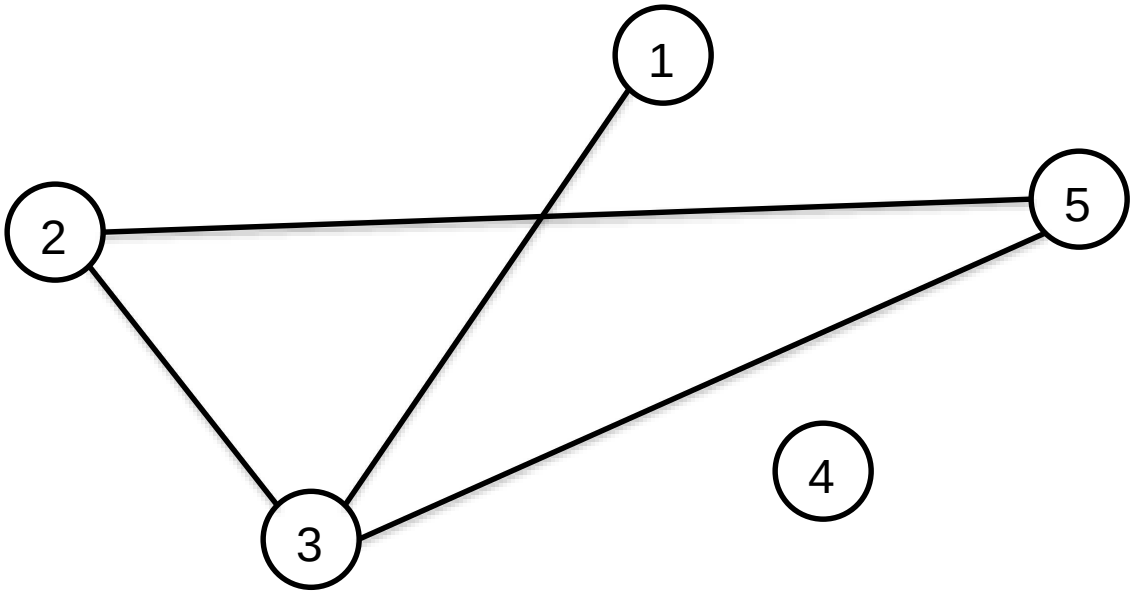
*Se numeşte **graf neorientat** o pereche ordonată de mulţimi **(X,U)**, **X** fiind o mulţime finită şi nevidă de elemente numite **noduri** sau **vârfuri**, iar **U** o mulţime de perechi neordonate din **X**, numite **muchii**.*

6.1. Definiţii

Exemplu:

$X = \{1, 2, 3, 4, 5\};$

$U = \{(1,3), (2,3), (2,5), (3,5)\};$



Definiţie:

- Două vârfuri legate printr-o muchie se numesc adiacente.

Definiţie:

- Muchia [x,y] este incidentă cu vârful x și vârful y.

Definiţie:

- Se numește gradul unui vârf x și se notează d(x) numărul vârfurilor adiacente cu vârful x.

Observaţie:

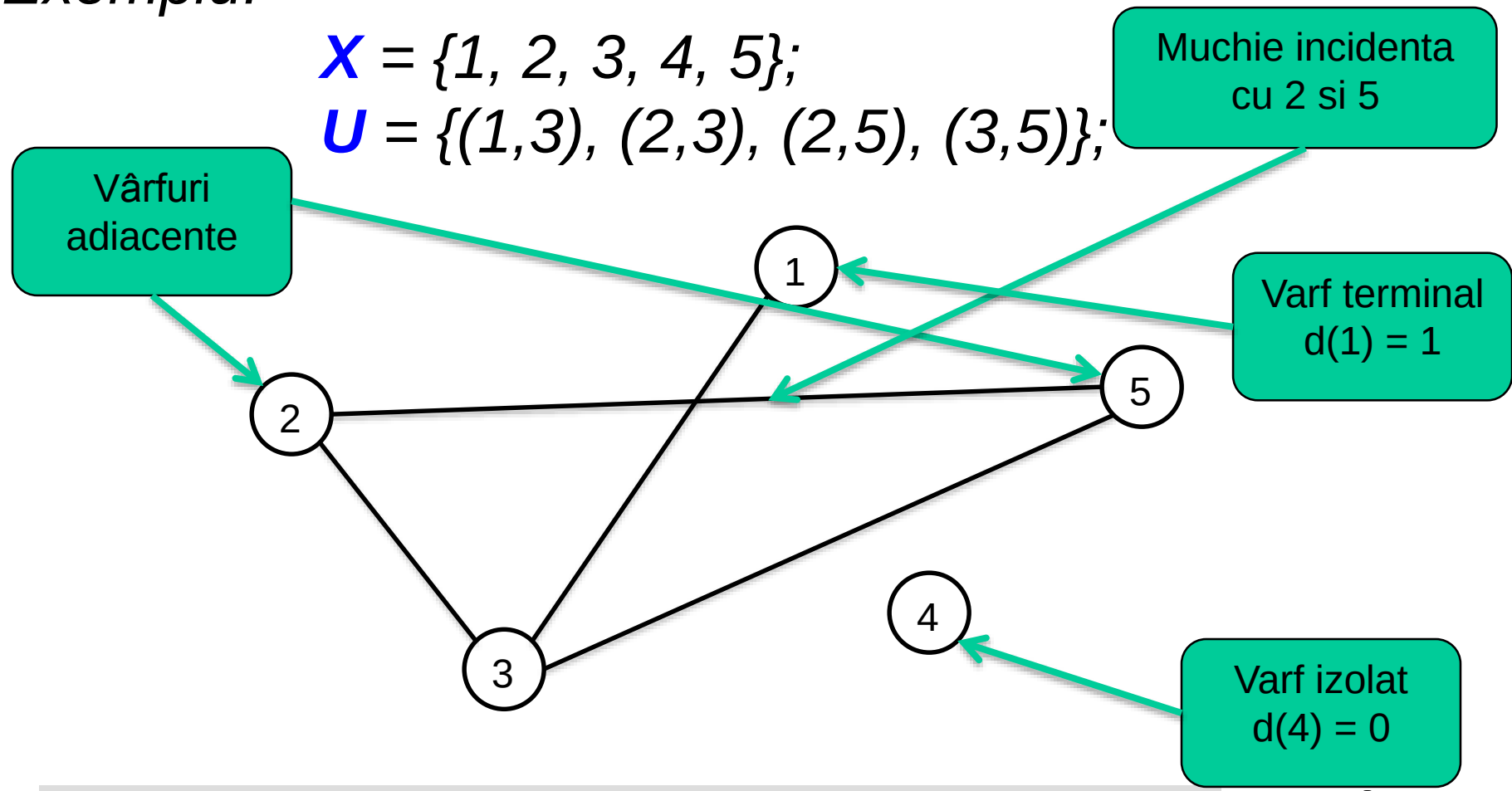
- Vârfurile cu grad 0 se numesc izolate
- Vârfurile cu grad 1 se numesc terminale

6.1. Definiţii

Exemplu:

$X = \{1, 2, 3, 4, 5\};$

$U = \{(1,3), (2,3), (2,5), (3,5)\};$



6.1. Definiţii

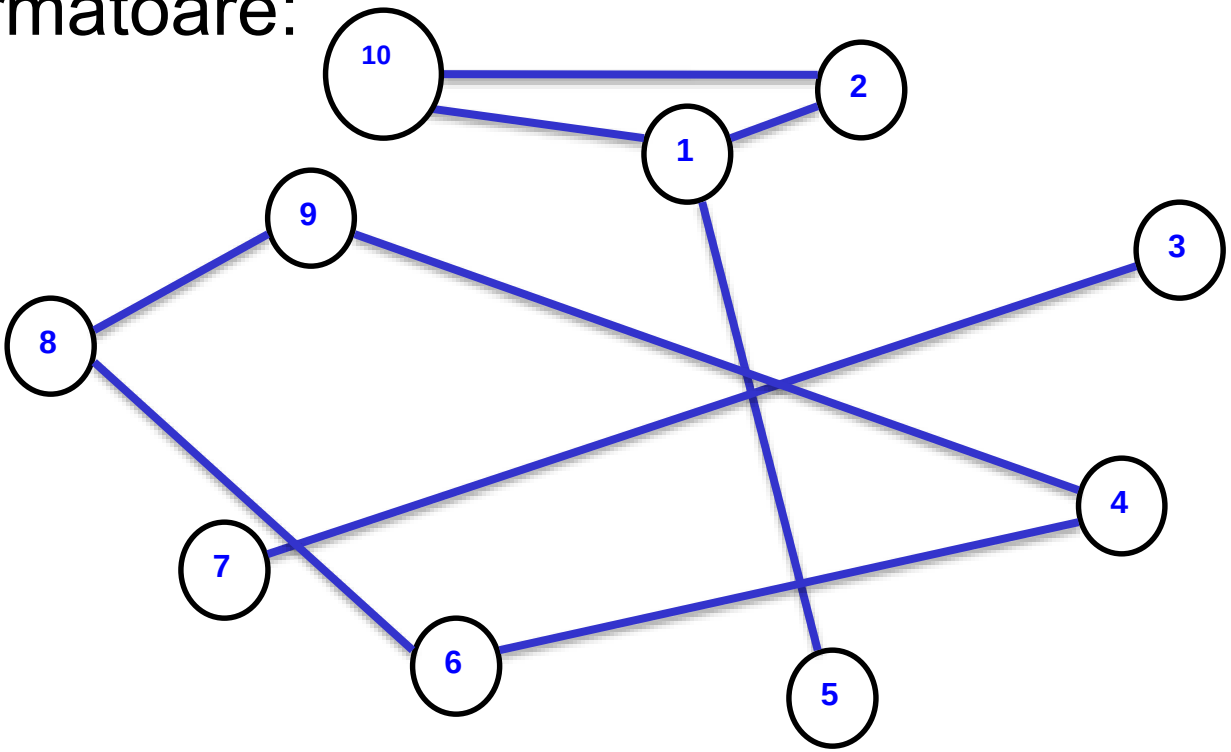
- Un prim element atrăgător al teoriei grafurilor este *aspectul geometric sau grafic* al subiectelor.
- Astfel *un graf poate fi reprezentat cu ajutorul unei figuri plane in care fiecărui vârf i se asociază un punct şi fiecărui muchii(x,y) o linie curbă care uneşte în plan punctele ce corespund vârfurilor x şi y*.
- Al doilea aspect interesant este *dezvoltarea naturală a teoriei grafurilor*, de îndată ce definiţia unui graf a fost prezentată, noţiunile şi rezultatele par să se nască singure şi în mod spontan, astfel încât cel care studiază acest domeniu pare să aibă impresia că ar fi putut fi chiar el însuşi creatorul acestui domeniu.

Exemplu

Fie G=(X,U) astfel încât X=(1,2,3,4,5,6,7,8,9,10)

U={(1,5),(3,7),(4,6),(9,8),(10,2),(1,2),(9,4),(1,10),(6,8)}.

Reprezentarea în plan a acestui graf este dată în figura următoare:



6.1. Definiţii

TEOREMA:

In orice graf (X,U) suma gradelor varfurilor este de doua ori numarul de muchii, adica

$\sum d(x) = d(x_1) + d(x_2) + \ldots + d(x_n) = 2 * m,\ x \in X,$

Unde: m – nr. de muchii, n – nr. de varfuri.

Demonstratie:

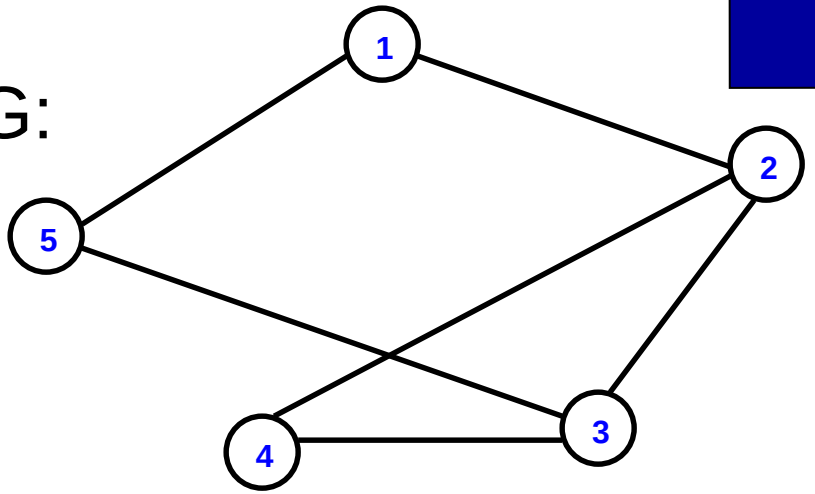
Fiecare muchie (x,y) contribuie cu o unitate la gradul lui x si cu o unitate la gradul lui y, deci contribuie cu 2 unitati la suma gradelor, si atunci fiind m muchii inseamna ca suma gradelor este de 2 ori nr. de muchii.

6.1. Definiţii

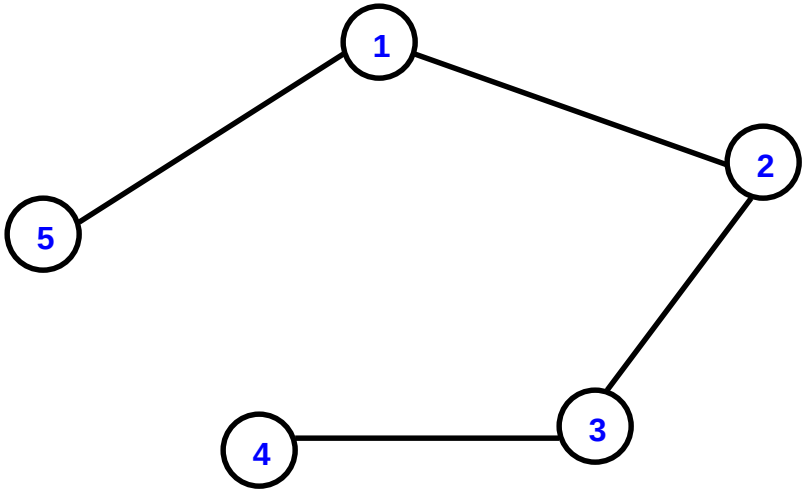
Definiţie

- *Un **graf parţial** al grafului G=(X,U) este un graf G_1=(X,V) astfel încât $V \subseteq U$, adică G_1 are aceeaşi mulţime de vârfuri ca G iar mulţimea de muchii V este chiar U sau o submulţime a acesteia.*
- Cu alte cuvinte, *un graf parţial al unui graf se obţine păstrând aceeaşi mulţime de vârfuri şi eliminând o parte din muchii*.
- Se mai spune că un graf parţial G_1=(X,V) este indus de mulţimea V de muchii.

Exemplu: Fie graful G:



Atunci graful G_1 este **graf partial** pentru graful G



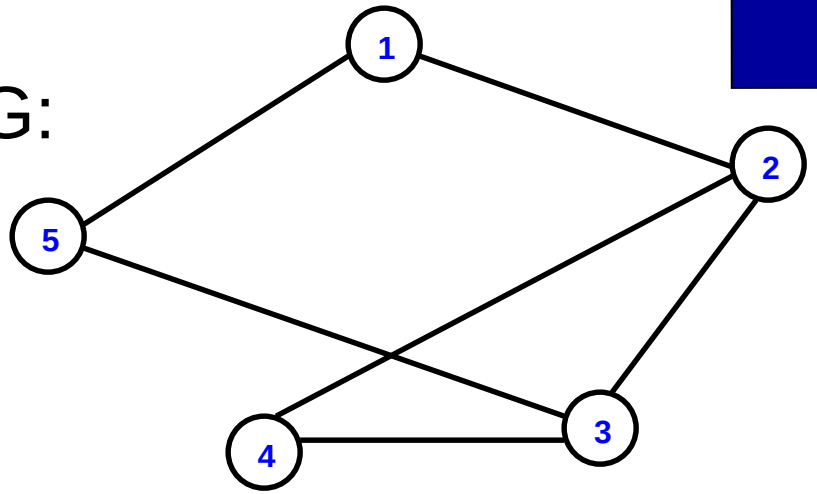
6.1. Definiţii

Definiţie

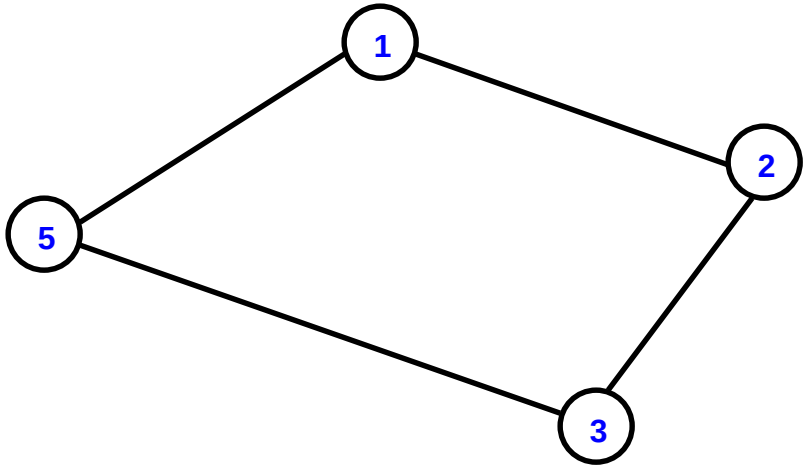
- *Un **subgraf** al unui graf G=(X,U) este un graf H=(Y,V) astfel încât $Y \subset X$ iar V conţine toate muchiile din U care au ambele extremităţi in Y.*

- Prin urmare, *un subgraf al unui graf se obţine eliminând o parte din vârfuri şi toate muchiile incidente cu acestea*.

Exemplu: Fie graful G:



Atunci graful G_1 este **subgraf** pentru graful G



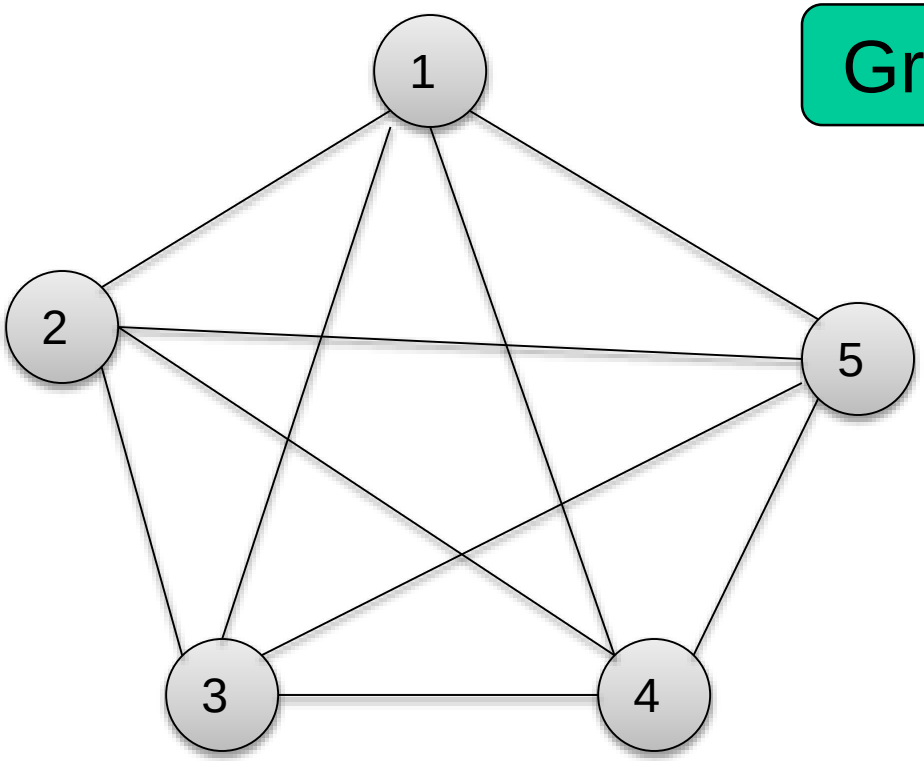
- Există tipuri de grafuri care au proprietăţi speciale şi care intervin în anumite categorii de probleme şi raţionamente.
- Ele au denumiri şi notaţii speciale.

Definiţie

*Se numeşte **graf complet** cu n vârfuri, un graf care are proprietatea că orice două noduri diferite sunt adiacente.*

6.1. Definiţii

Graf complet



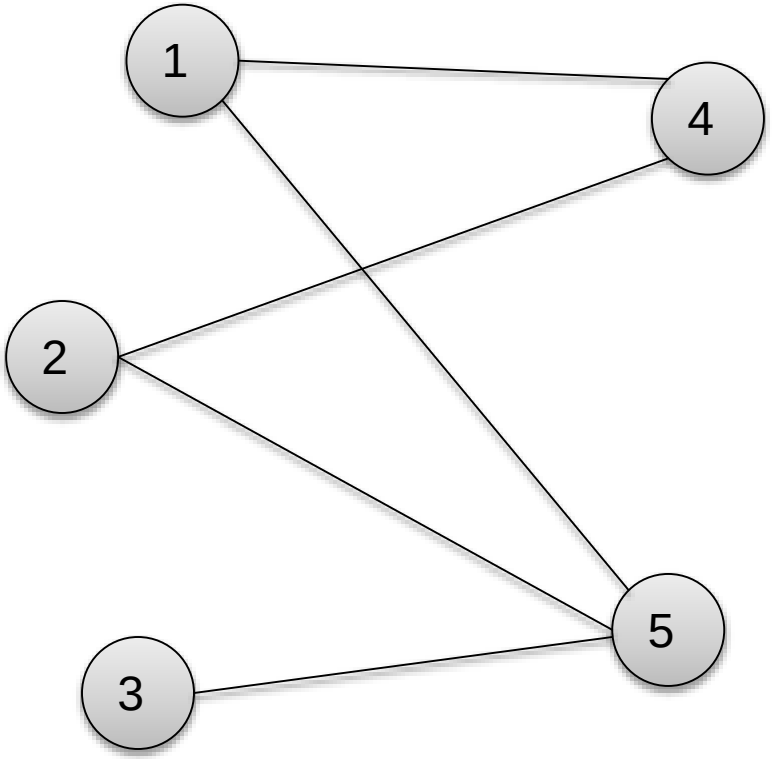
Definiţie

*Un graf G=(X,U) se numeşte **bipartit,** dacă există două mulţimi nevide A şi B astfel încât X=A∩B, şi orice muchie u a lui G are o extremitate în A iar cealaltă în B.*

Definiţie

*Un **graf bipartit** se numeşte **complet,** dacă pentru orice x din A şi orice y din B, există în G muchia [x,y].*

6.1. Definiţii



Graf bipartit

Definitie

Se numeste **lanț** în graful G, o succesiune de vârfuri $L=(x_1,x_2,x_3,...,x_n)$, unde $x_1, x_2, x_3, ..., x_n \in X$, cu proprietatea că oricare două vârfuri consecutive sunt adiacente, adică există muchiile $[x_{i-1},x_i]$, cu $i > 1$ și $i < n$.

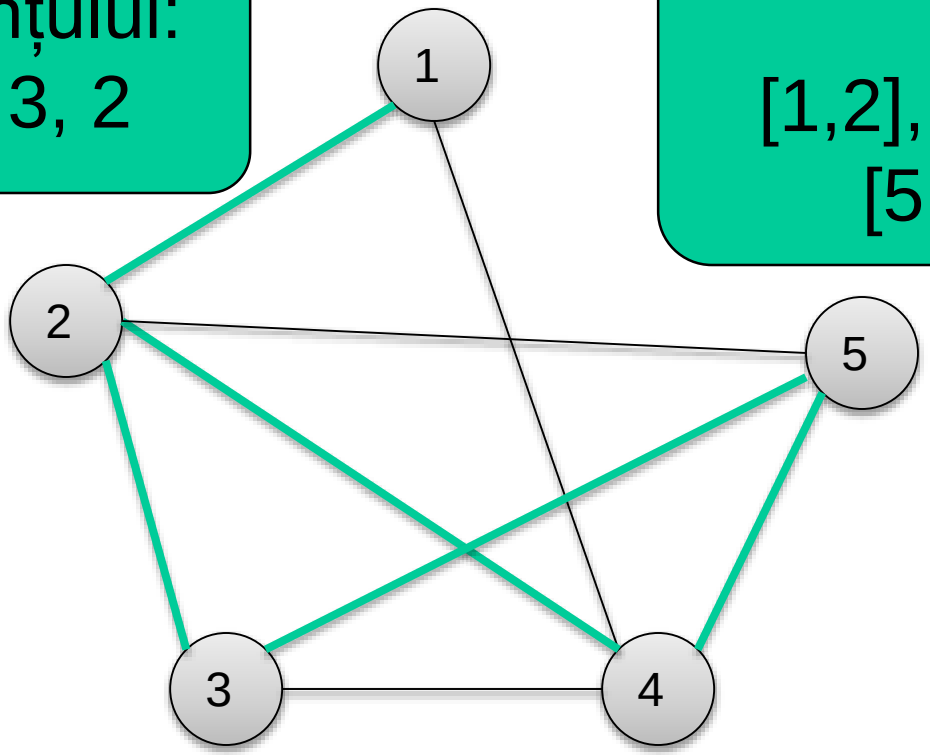
Observație:

Vârfurile x_1 și x_n se numesc extremitățile lanțului, iar numărul de muchii care intră în componența lanțului reprezintă lungimea lanțului.

6.1. Definiţii

Vârfurile lanțului:
1, 2, 4, 5, 3, 2

Lanț
[1,2], [2,4], [4,5],
[5,3], [3,2]



Lungimea lanțului: 5

Observație:

Dacă vârfurile $x_1, x_2, ..., x_n$ sunt distincte două câte două lanțul se numește elementar, în caz contrar se numește ne-elementar.

Observație:

Un graf este **conex** dacă între oricare două vârfuri ale sale există un lanț care le leagă.

6.1. Definiţii

Definiție

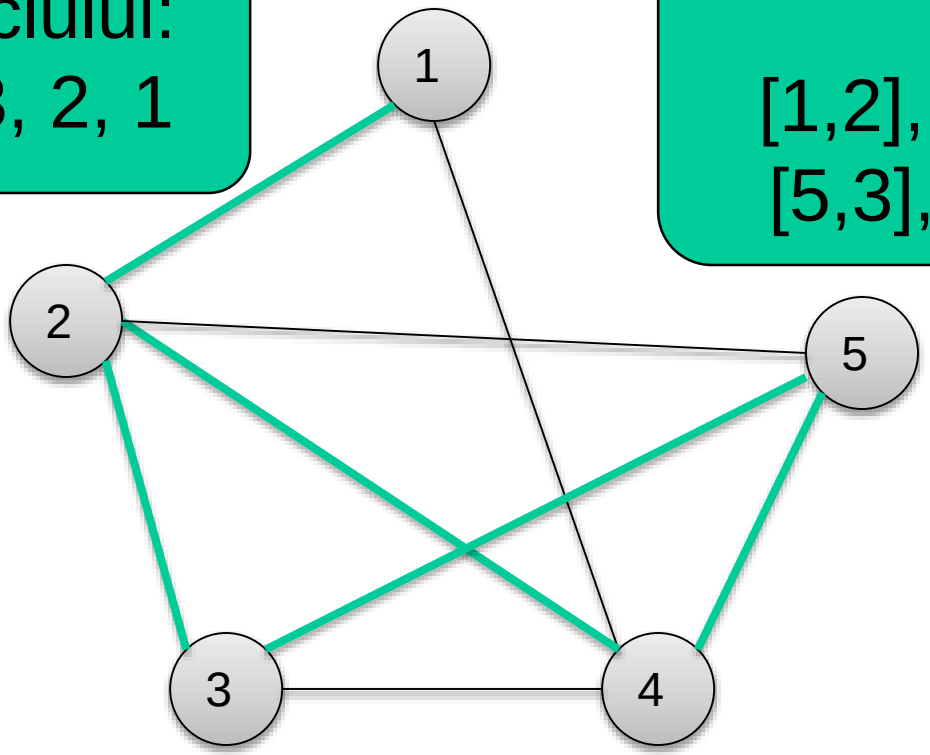
Se numeste **ciclu** într-un graf un lanț $L=(x_1, x_2, x_3, ..., x_n)$ cu proprietatea că $x_1 = x_n$ și că muchiile $[x_{i-1}, x_i]$ sunt distincte două câte două.

Observație:

Dacă vârfurile într-un ciclu cu excepția primului și ultimului sunt distincte două câte două, atunci ciclul se numește elementar, altfel ne-elementar.

6.1. Definiţii

Vârfurile ciclului:
1, 2, 4, 5, 3, 2, 1



Ciclu:
[1,2], [2,4], [4,5],
[5,3], [3,2], [2,1]

Conţinutul cursului

6.1. Definiţii

6.2. Memorarea(reprezentarea) grafurilor

6.3. Parcurgerea grafurilor

6.2. Memorarea(reprezentarea) grafurilor

- Fie G=(X,U) un graf neorientat.
- Deoarece între mulţimea X cu n elemente şi mulţimea {1,2,...,n} există o legătură directă, atunci există mai multe modalităţi de a reprezenta graful G, pentru care se folosesc diverse structuri de date.

- *Reprezentările vor fi utilizate în algoritmii care prelucrează grafuri, deci şi în programele care implementează pe calculator aceiaşi algoritmi.*

6.2. Memorarea(reprezentarea) grafurilor

Selectarea uneia sau a alteia dintre structurile de date ce vor fi prezentate este în funcţie de problema şi de algoritmul ales pentru rezolvarea ei:

1) Se precizează numărul *n* de vârfuri şi **matricea de adiacenţă** A a grafului, care este o matrice pătratică de ordinul n.
2) Se precizează numărul n de vârfuri şi, pentru fiecare vârf i, lista L_i a vecinilor săi, adică lista vârfurilor j pentru care [i,j] $\in$ U.

6.2. Memorarea(reprezentarea) grafurilor

3) Se dau numărul n de vârfuri, numărul m de muchii, precum şi două tablouri unidimensionale e_1 şi e_2 cu câte m componente fiecare, conţinând extremităţile muchiilor grafului, adică

$U=\{ (e_1[1],e_2[1]), (e_1[2], e_2[2]), ..., (e_1[m],e_2[m]) \}$

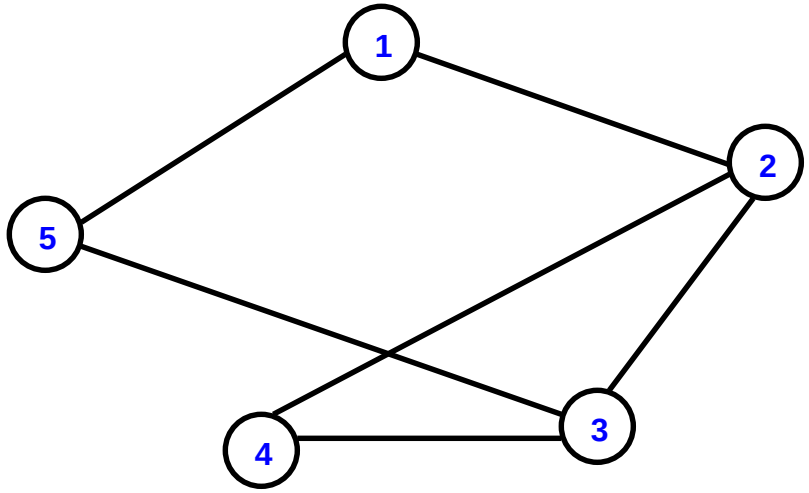
O variantă de implementare mai naturală ar fi aceea de a defini un tip de dată *muchie* utilizând tipul **struct**, care să conţină cele două extremităţi ale unei muchii, şi apoi de a defini un tablou cu n componente de acest tip:

```
typedef struct muchie {
        int x, y;
 }
  .....
  struct muchie u[20];
```

- Referirea la extremităţile muchiei i se face prin:
  - u[i].x
  - respectiv u[i].y
- Acest mod de reprezentare permite înglobarea naturală în tipul de date *muchie* şi a altor infomaţii asociate muchiilor şi este utilizată în problemele în care muchiile se prelucrează succesiv, eventual după o modificare a ordinii lor în tabloul u.

1) Reprezentarea grafurilor folosind matricea de adiacență:

Fie graful urmator:



Graful are n=5 vârfuri

Matricea de adiacenta corespunzatoare grafului este:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Citirea unui graf memorat cu ajutorul matricei de adiacență

Varianta 1:

Citirea numarului de varfuri, citirea regiunii de deasupra diagonalei principale urmata de **simetrizare**.

```
#include<iostream.h>
int main(void)
{
    int a[20][20], i, j, n;
    cout<<“nr. de varfuri = ”; cin>>n;
    for(i=1; i<=n-1; i++)
        for(j=i+1; j<=n; j++){
            cout<<“a[”<<i<<“][”<<j<<“]=”;
            cin>>a[i][j];
            a[j][i] = a[i][j];
        }
}
```

Varianta 2:

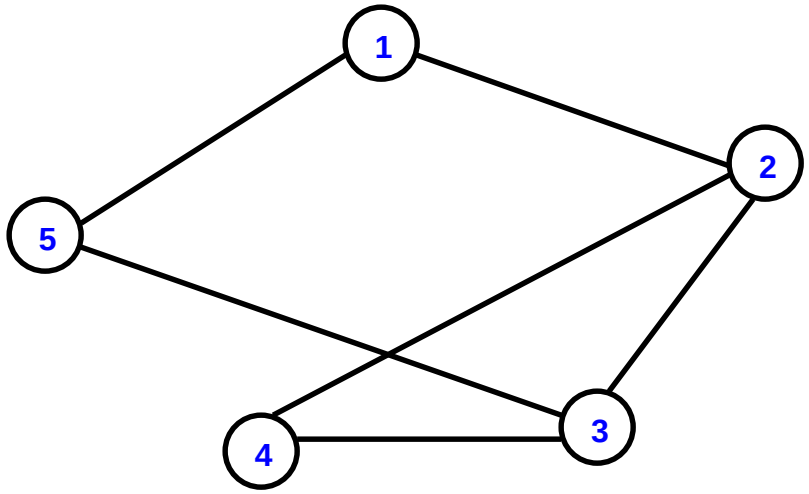
Se citesc n - nr de varfuri, m - nr de muchii;

- initializarea matricei de adiacenta cu zero
- citirea extremitatii fiecarei muchii
- completarea matricei de adiacenta

```
#include<iostream.h>
int main(void)
{
    int a[20[20], i, j, n, m;
    cout<<“nr. de varfuri = ”; cin>>n;
    cout<<“nr. de muchii = ”; cin>>m;
    for(i=1; i<=n-1; i++)
            for(j=i+1; j<=n; j++) a[i][j] = 0;
    for(i=1; i<=m; i++)
    {
            cout<<“Muchia ”<<i<<“ este ”;
            cin>>x; cin>>y;
            a[x][y] = 1;
            a[y][x] = 1;
    }
}
```

2) Reprezentarea grafurilor folosind listele de vecini ale fiecarui varf:

Fie graful urmator:



Graful are n=5 vârfuri

Listele de vecini corespunzatoare grafului sunt:

Nodul 1: are vecini pe 2,5

Nodul 2: are vecini pe 1,3,4

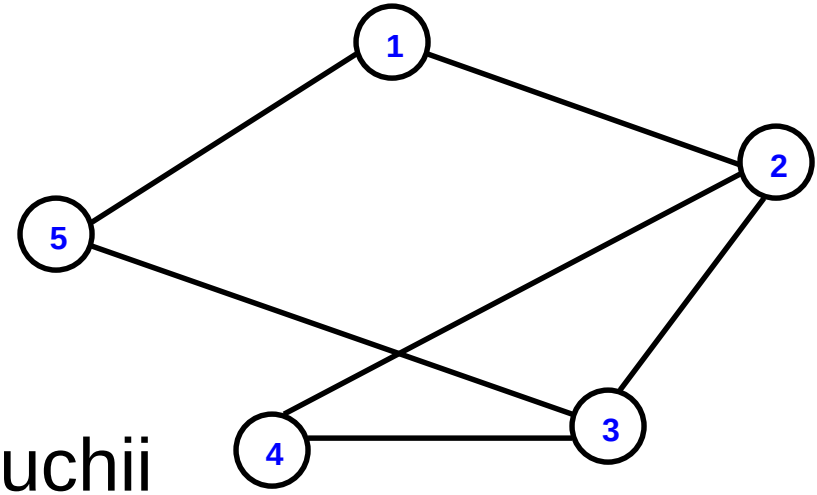
Nodul 3: are vecini pe 2,4,5

Nodul 4: are vecini pe 2,3

Nodul 5: are vecini pe 1,3

3) Reprezentarea grafurilor folosind doua tablouri unidimensionale in care se retin extremitatile fiecarei muchii din graf:

Fie graful urmator:



Graful are n=5 vârfuri si m=6 muchii

Cele tablouri contin elementele:

e_1 = {1, 1, 2, 2, 3, 3}

e_2 = {2, 5, 3, 4, 4, 5}

Atunci graful este format din multimea de muchii:

U = { (1,2), (1,5), (2,3), (2,4), (3,4), (3,5) }

Conţinutul cursului

6.1. Definiţii
6.2. Memorarea(reprezentarea) grafurilor
6.3. Parcurgerea grafurilor

Probleme cu grafuri neorientate

1. Memorarea grafurilor neorientate folosind cele trei modalitati de reprezentare:

a) Matrice de adiacenta

b) Vector de muchii

c) Liste de vecini

Probleme cu grafuri neorientate

Solutie

```
#include <fstream.h>
#include <vector.h>
ifstream fin("date.in");
ofstream fout("date.out");
struct muchie{
    int i,j;
};
int A[50][50];          //      matrice de adiacenta
muchie M[1000];         //      vector muchiilor
int V1[50], V2[50];     //      liste de vecini
int n,m;                //      n – nr. de noduri, m – nr. de muchii
```

Probleme cu grafuri neorientate

Solutie (continuare)

```
void citire()
{
   int x,y,i;
   fin>>n>>m;
   for(i=1;i<=m;i++)
   {
      fin>>x>>y;
      M[i].i=x; M[i].j=y;
      A[x][y]=A[y][x]=1;
      V1[i]=x;
      V2[i]=y;
   }
}
```

Probleme cu grafuri neorientate

Solutie (continuare)

```
void afisare()
{
    int i,j;
    fout<<"matricea de adiacenta:\n";
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        fout<<A[i][j]<<" ";
        fout<<endl;
    }
    fout<<"lista muchiilor:\n";
    for(i=1;i<=m;i++)
      fout<<M[i].i<<" "<<M[i].j<<endl;
    fout<<"lista vecinilor:\n";
    for(i=1;i<=n;i++)
    {
        fout<<i<<": ";
        fout<<V1[i]<<","<<V2[i]<< " ";
        fout<<endl;
    }
    fin.close();
    fout.close();
}
```

Probleme cu grafuri neorientate

Solutie (continuare)

```
int main()
{
    citire();
    afisare();
}
```

Probleme cu grafuri neorientate

2. Se citeste dintr-un fisier de pe primul rand o valoare n reprezentand numarul de noduri pentru un graf neorientat si de pe urmatoarele n linii si coloane matricea de adiacenta corespunzatoare grafului.

a) Identificati multimea X
b) Identificati multimea U
c) Calculati gradele nodurilor impare
d) Verificati daca graful are varfuri izolate, daca da afisati nodul, daca nu afisati un mesaj

Întrebări?